

WaterlooClarke: TREC 2015 LiveQA Track

Alexandra Vtyurina
University of Waterloo
avtyurina@uwaterloo.ca

Ankita Dey
University of Waterloo
a5dey@uwaterloo.ca

Bahareh Sarrafzadeh
University of Waterloo
bsarrafz@uwaterloo.ca

Charles L. A. Clarke
University of Waterloo
claclark@plg.uwaterloo.ca

ABSTRACT

The goal of the LiveQA track is to automatically provide answers to questions posted by real people. Previous question answering tracks included factoid questions, list questions and complex questions[3]. Presented in 2015 for the first time the LiveQA track gave the participants an opportunity to answer questions posed by real people, as opposed to manually configured ones in the previous tasks.

The questions for the task were harvested from Yahoo! Answers¹ – a community question answering website. Each question was broadcasted to all registered systems. The participants had to supposed to provide an answer to every given question within a timeframe of 60 seconds. The answers were judged by human NIST assessors after the evaluation was over.

1. INTRODUCTION

The task of automatic question answering has appeared a multiple times in TREC. The tracks have moved from answering factoid questions to list questions, and questions with complex information need. Although the questions were modelled to imitate human askers, they were not coming from real people and the answers often had to be extracted from a restricted corpora of newswire and blogposts.

LiveQA track brought the task to the new level by providing real world questions, unlimited corpora usage and restricting the answer time. The questions for this task were coming from Yahoo! Answers – a community question answering website. Questions there vary greatly between all topics and question types. Yahoo! Answers users are often seeking other people's opinion, an advice about a problem they're having. Some of them just want to share their insights or emotions about newly acquired knowledge or experience. In certain cases people do not have a well defined information need, but they are looking to start a conversation (see Table

¹<https://answers.yahoo.com>

Title: Is my nose too big?
Body: Is my nose bad or horrible. I know it's bigger but just how bad is it
Title: Whats the meaning of life?
Body: i wanna know YOUR meaning of life!
Title: Emma Stone, Mila Kunis or Penelope Cruz? Who's most beautiful in your opinion?
Body: I can't decide they are all gorgeoussss <3 :)

Table 1: Various types of questions asked on Yahoo! Answers

1).

The questions were collected from the list of newly posted (and not yet answered by human users) questions on Yahoo! Answers. Each question was sent to every participating system and an answer was expected to arrive within a 60-seconds window. The answer was supposed to contain, among other fields, a text snippet of length less than 1000 characters and the list of resources, from which it was obtained. If the answer was received after 1 minute, it did not count towards the total score of the system. Participants could also choose not to answer any question.

The evaluation of the answers given by participating systems was done by human NIST assessors on a 5-level Likert scale.

2. EXPERIMENTAL SETUP

The experiment was running for the duration of 24 hours starting at 12am PST August 31, until 12am PST on September 1. During this period of time the participating systems were supposed to be online. Yahoo! server collected newly posted question from Yahoo! Answers and broadcasted it to all the registered systems at a rate of approximately 1 question per 60 seconds.

Every question consisted of 4 fields: *qid* - question identifier, *title* - a question, formulated by a person, *body* - optional detailed description of the question, and finally, *category* - the category that the person chose for their question (if the user skips the step of picking a category, it is defined automatically by Yahoo! Answers).

A response to each question was expected upon 60 seconds after sending. It was supposed to contain the following fields: *pid* - participant id (`uwaterlooclarke` was used for this sub-

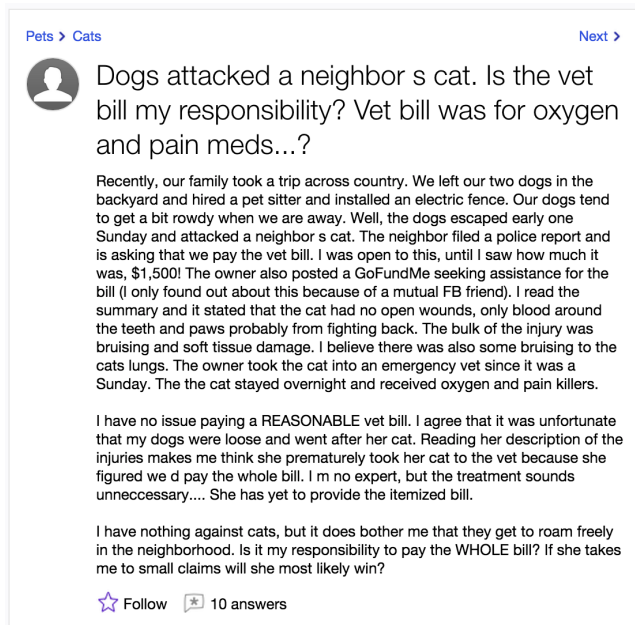


Figure 1: An example of a long question, containing a lot of detailed information

mission), *qid* - question identifier, *answer* - a text of length of max 1000 characters, *sources* - a list of sources where the answer was fetched from, *local time* - locally measured time in ms it took to produce the answer, *explanation* - an optional string containing additional information about the answer. Responses that were received after the 60 seconds were not judged.

3. GENERAL APPROACH

Our approach was based on finding an answer on the internet using a search engine. After receiving a question we picked key terms from its text and constructed a query. We then used the query to retrieve a set of top-ranked web documents from Bing. Afterwards, we used the obtained set of documents to extract passages that were likely to answer the question. Finally, we ranked all the passages and returned the highest-ranked one as an answer to the given question.

3.1 Background model

In order to use KL-divergence for query term extraction we needed to have a background language model. Given that the language used in online user-generated content differs significantly from formal English [1], we needed to have an example of the language that is usually used on Yahoo! Answers.

We crawled Yahoo! Answers to collect a dataset of questions and answers from all categories in order to see what type of language is used by people there²³. For each question thread we collected question title, question body, and answers (if any), posted by other people. We removed web

links from obtained text and used the rest of the text to build a language model.

3.2 Answer extraction

For every question received we combined its *title* and *body* together and removed the links from the resulting text snippet. We compared the words distributions in the question text and the previously constructed language model and picked the words with the greater divergence value, which means that these words distinct the given question from the common language. For every word in the text a corresponding KL-divergence[2] value was computed, using the Yahoo! Answers language model constructed earlier. Afterwards, the words were sorted based on their corresponding KLD score. We also used NLTK⁴ to extract named entities from the question text. These named entities as well as the 4 words with the highest KLD score were put together in the order of their occurrence in the initial question to form a resulting query.

The query was submitted to the Bing Search API and the top-10 returned documents were retrieved. We ignored pages from Yahoo! Answers, as well as all non-html pages (for example, pdf). For every web-page we allowed a 5 seconds time limit to load, otherwise it was ignored. The response from Bing came in json format and contained *description* - a short text snippet extracted from a document, and the document's *url*. We used this set of web documents as a corpus to extract an answer to the given question from.

After the web pages are retrieved, they undergo a preprocessing step, during which only useful text was extracted from each of them. First, we removed the contents of a predefined list of tags (that are highly unlikely to contain the useful text that we are after): style, script, table, label, title, etc. From the remaining portion of the page the tags, with contents of less than 10 words are removed. By doing this we excluded ads, "follow us" links, and other irrelevant information.

After the preprocessing every web page becomes a clean raw text. At this step we insert a pair of special symbols used to denote the beginning and the end of each sentence. This is done in order to produce more readable results in the future. For every document we found a set of m-covers (passages containing keywords), using the terms from the query we previously submitted to Bing. If the length of a passage was greater than the given limit (1000 characters), it was discarded. The remaining passages were ranked according to the number of query terms they contained and their proximity to each other within the passage[2]. After the passages were scored, the highest-ranked one was considered to be the answer. At this point the borders of the passage were stretched to the closest beginning and end of a sentence. The URL, corresponding to was final passage is passed along as the resource of the answer.

4. CODE BASE

The primary module (Java module) for communication with Yahoo! server was supplied by the track organizers⁵. We

²All code used for this task is available at:
<https://github.com/sashavtyurina/LiveQATrack>

³<https://github.com/yuvalpinter/LiveQAServerDemo>

⁴<http://nltk.org/>

⁵<https://github.com/yuvalpinter/LiveQAServerDemo>

used a separate module written in Python to process incoming questions and extracting answers. The two modules communicated with each other using Twisted⁶ networking library by sending to each other messages in json format.

5. FUTURE WORK

We would like to improve the procedure of finding answer to a given question by analysing existing human-generated question-answer pairs. We are hopeful that finding the ways in which an answer is related to the question will help extract more precise answers in the future.

It is not uncommon for community question answering services to have an exceedingly long question descriptions. People often want to see an advice that is unique for their situation (see figure 1). Redundant details often obstruct question focus, making it hard even for a human to understand. We want to reduce such long questions to a length of 2-3 sentences by extracting only the sentences, reflecting the user's information need.

6. CONCLUSIONS

The LiveQA track revives the task of automatic question answering in TREC. It provides an opportunity for the participants to try their QA systems on real-world questions, collected from Yahoo! Answers – community question answering website. The approach we chose is based on picking key terms from a given question, submitting them to a search engine and extracting an answer from the top 10 retrieved documents.

7. REFERENCES

- [1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 183–194. ACM, 2008.
- [2] S. Büttcher, C. L. Clarke, and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2010.
- [3] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, page 63. Citeseer, 2007.

⁶<https://twistedmatrix.com/>